December 12, 2001
Q67327AP.BCR

METHOD OF UPGRADING SOFTWARE IN A NETWORK ENVIRONMENT AND A NETWORK DEVICE FOR PERFORMING THE SAME

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method of upgrading software in a network environment and a network device for performing the upgrade, and more particularly, to a method of upgrading software, through which a network device can be protected when an unexpected failure occurs in a network or a network device during an upgrade of the software, and a network device for performing the same. The present application is based on Korean Application No. 2001-38804, filed June 30, 2001, which is incorporated herein by reference.

2. Description of the Related Art

As network environments develop, the need of network service providers, network operators, and network device users to upgrade software for network devices increases. Network devices include mobile and portable telephones and personal computers that can transmit and receive data through a network.

However, when an unexpected failure occurs in a network device or a network during a software upgrade of the network device, the upgrade is not completed normally. Moreover, a catastrophic influence may be exerted on the network device.

In other words, when a failure occurs in a network while a new version of software is being downloaded through the network to upgrade the software of a network device, the new version of the software cannot be downloaded normally. Here, the problem can be overcome by trying to download the new version again.

However, when an unexpected failure occurs in a network device from which the old version of the software has been erased, while a new version of the software is being stored in a storage area to upgrade the software, the network device cannot normally store the new version of the software. As a result, the network device cannot perform functions that use the software.

In the case where the software to be upgraded does not greatly affect the operation of a network device, even when an unexpected failure occurs in the network device as described above, a problem can be prevented by connecting an interface block provided in the network device to an external memory device and newly downloading the software. However, in this case, it is inefficient that the external memory device must be managed to store the new version of the software or to store the version of the software which is currently stored in the network device.

In the case where the software to be upgraded greatly affects the operation of a network device, as in an operating system (OS), when an unexpected failure occurs in the network device as described above, the network device is catastrophically affected and does not operate at all. Here, the problem cannot be solved solely by using an external memory device, as described above. A corresponding chip or the network device must be replaced.

SUMMARY OF THE INVENTION

To solve the above-described problems, it is a first object of the present invention to provide a method of upgrading software, through which the operation of a network device is not catastrophically affected by a failure occurring while software is being upgraded in a network environment, and a network device for performing the same.

It is a second object of the present invention to provide a method of upgrading software, through which a network device can operate normally even if a software upgrade is not accomplished normally in a networked environment due to a failure, and a network device for performing the same.

To achieve the above objects of the invention, there is provided a method for upgrading software of a network device through a network. The method includes the steps of upgrading software through the network and checking whether at least one failure occurs during the upgrade, operating the network device based on an old version of the software used before the upgrade is performed when it is determined that at least one failure has occurred, and operating the network device based on a new version of the software to which the old version is upgraded when it is determined that a failure has not occurred.

Preferably, the upgrading of software includes the steps of downloading the new version of the software through the network, copying the old version of the software stored in a first area of the network device to a second area of the network device, erasing the old version of the software from the first area of the network device, and storing the new version of the software in the first area.

Preferably, the failure is a failure in the network device which is checked during the erasing and storing steps.

To achieve the above objects of the invention in one embodiment, there is provided a network device capable of upgrading software through a network. The network device includes a monitoring unit for monitoring at least one failure while software is being upgraded; a first memory for storing software necessary for operating the network device; a second memory for storing information transferred through the network; a controller for performing control to store information, which is downloaded through the network to upgrade the software, in the second memory, and store an old version of the software in an empty area of the first memory before the old version of the software stored in the first memory is upgraded with the information stored in the second memory; and a decoder for selecting a memory, which is used for upgrading the software, from the first memory and the second memory according to a control signal received from the controller and the result of monitoring received from the monitoring unit, and setting an address.

In another embodiment, there is provided a network device capable of upgrading software through a network. The network device includes a monitoring unit for monitoring whether at least one failure occurs while software is being upgraded; a first memory for storing software necessary for operating the network device; a second memory for storing data necessary for operating the network device; a third memory for storing information transferred through the network; a controller for performing control to store information, which is downloaded through the network to upgrade the software, in the third memory, and store an old version of the software in an empty area of the second memory before the old version of the software stored in the first memory is upgraded to the information stored in the third memory; and a decoder for selecting a memory, which is used for upgrading the software, according to a control signal received from the controller and the result of monitoring received from the monitoring unit, and setting an address.

BRIEF DESCRIPTION OF THE DRAWINGS

The above objects and advantages of the present invention will become more apparent by describing in detail preferred embodiments thereof with reference to the attached drawings in which:

FIG. 1 is a block diagram of a network device according to a preferred embodiment of the present invention; and

FIG. 2 is a flowchart of a method of upgrading software according to a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 1, a network device according to the present invention includes a monitoring unit 100, a network interface unit 104, a chip selection/address decoder 105, a controller 106, a conditional access system (CAS) 107, a flash memory 108 for storing code data, a non-volatile memory 109 for storing data, and a system memory 110 for storing execution data.

The monitoring unit 100 monitors at least one failure while software of a network device is being upgraded. The monitoring unit 100 includes a watchdog monitor 101, a power failure monitor 102, and a network connection monitor 103.

The watchdog monitor 101 monitors generation of a clock signal through which the operation of the network device can be monitored to monitor whether the network device operates normally, and sends the monitoring results to the chip selection/address decoder 105.

The power failure monitor 102 monitors whether a power voltage supplied from a power supply (not shown) within the network device drops below a predetermined level, and sends the monitoring result to the chip selection/address decoder 105.

The network connection monitor 103 monitors the state of data transmitted and received through the network interface unit 104, and sends the monitoring results to the chip selection/address decoder 105.

The network interface unit 104 performs interfacing to allow data communications between a party, which provides the new version of the software to be upgraded through a network (not shown), and the network device shown in FIG. 1. In particular, the network interface unit 104 can be realized by decoding received software and checking received data for errors. Data error checking can be performed by way of checksum.

The CAS 107 transmits data and receives data between the party providing software through the network and the network device shown in FIG. 1 and performs authentication of the network device for a software upgrade. In other words, the CAS 107 verifies whether the network device has the authority to upgrade the corresponding software. Here, upgrading of software can be performed according to the determination of a software provider or at the request of a network device user.

The flash memory 108 for code data stores software such as an operating system (OS) necessary for operating the network device. When software stored in the flash memory 108 is upgraded, if the flash memory 108 has an empty area, the old version of the software can be stored in the empty area through the operation of the chip selection/address decoder 105.

The non-volatile memory 109 for data stores data necessary for operating the network device. In particular, when software stored in the flash memory 108 is upgraded, the old version of the software that has been stored in the flash memory 108 can be stored in an empty area existing in the non-volatile memory 109 through the operation of the chip selection/address decoder 105.

The system memory 110 stores information transmitted through the network when the network device is operated.

The controller 106 controls the network device to be able to upgrade software. In other words, once the new version of the software to be upgraded is downloaded through the network interface unit 104 and the CAS 107, the controller 106 sends a control signal to the chip selection/address decoder 105 so that the new version of the software can be stored in the system memory 110. Accordingly, the chip selection/address decoder 105 sets a chip selection signal for the system memory 110 and an address to indicate an area of the system memory 110 in which the software is to be stored so that the received new version of the software can be stored in a desired area of the system memory 110.

Next, the controller 106 sends a control signal to the chip selection/address decoder 105 so that the old version of the software stored in the flash memory 108 can be stored in an empty area of the flash memory 108 or in an empty area of the non-volatile memory 109. Accordingly, when the old version of the software is stored in the empty area of the flash memory 108, the chip selection/address decoder 105 sets a chip selection signal and an address so that a path for storing can be set. In other words, the chip selection/address decoder 105 does not change the chip selection signal but changes the address. When the old version of the software is stored in the empty area of the non-volatile memory 109, the chip selection/address decoder 105 sets a chip selection signal and an address so that a path for storing can be set. In other words, the chip selection/address decoder 105 changes both the chip selection signal and the address. Here, when the start address of the empty area of the non-volatile memory 109 is the same as the start address of an area where the old version of the software is stored in the flash memory 108, the chip selection/address decoder 105 changes only the chip selection signal. Therefore, the old version of the software stored in the flash memory 108 is copied to another area.

The controller 106 also sends a control signal to the chip selection/address decoder 105 so that the new version of the software stored in the system memory 110 can be stored in an area where the old version of the software is stored in the flash memory 108, thereby upgrading the software. The area where the old version of the software is stored is an area where the original old version of the software is stored in the flash memory 108 before the copying is performed.

The chip selection/address decoder 105 sets a chip selection signal and an address in response to a control signal sent by the controller 106 so that the new version of the software stored in the system memory 110 can be stored in the flash memory 108.

While software is being upgraded, the chip selection/address decoder 105 continuously monitors the result of monitoring provided from the monitoring unit 100. When a signal indicating a failure is received from the network connection monitor 103 while the new version of the software is being downloaded to the system memory 110, the chip selection/address decoder 105 is initialized. When a signal indicating a failure is received from the watchdog monitor 101 or the power failure monitor 102 while information stored in the system memory 110 is being stored in the flash memory, the chip selection/address decoder 105 sets a chip selection signal and an address to select a memory necessary for restarting the network device based on the old version of the software.

FIG. 2 is a flowchart of a method of upgrading software according to an embodiment of the present invention. In an initial state in step 201, when the network connection between a software provider and a network device as shown in FIG. 1 is set in step 202, the version of software to be provided by the software provider is compared with the version of the software provided in the network device in step 203. If the version of the software provided in the network device is not old, upgrading of the software is not necessary and the operation goes back to step 201 so that the network device is maintained at the initial state.

If the version of the software provided in the network device is old, upgrading of the software is started in step 204. User authentication is performed in step 205. In other words, authentication is performed through data transmission between the software provider and the network device for user authentication, as described with respect to the CAS 107 of FIG. 1. The authentication is accomplished by a conventional method.

If the user authentication fails in step 205, the network device does not have the authority to upgrade the software and the operation goes back to step 201. In contrast, if the user authentication succeeds in step 205, the software is downloaded to the system memory 110 and a check is made as to whether a network disconnection has occurred and whether an error has occurred in the data of the received software in step 206. Data error checking for the received software can be performed by way of checksum.

When it is determined that a network disconnection or data error occurred in step 207, the operation goes back to step 201, and upgrading of software is not performed.

In contrast, when it is determined that neither a network disconnection nor a data error occurred in step 207, a check is made as to whether downloading is completed in step 208. If it is determined that downloading is not complete in step 208, steps 206 and 207 are repeated.

If it is determined that the downloading is completed in step 208, the old version of the software is copied to another area in step 209. Another area may be an empty area of the flash memory 108 or an empty area of the non-volatile memory 109, as described in FIG. 1.

In step 210, the software is upgraded to a new version, and it is checked whether a failure such as a power failure or system hang-up occurs in the network device during the upgrading. Here, for a software upgrade, a path to an area in which the old version of the software is stored is changed to a path to the area to which the old version has been copied. In other words, when the old version has been copied to an empty area of the flash memory 108, the chip selection/address decoder 105 sets a chip selection signal and an address to designate the empty area of the flash memory 108. Next, the original old version of the software is erased from the flash memory 108, and then the new version of the software stored in the system memory 110 is copied to the area from which the original old version was erased in the flash memory 108.

The monitoring unit 100 checks whether a failure such as a power failure or system hang-up occurs in the network device during the upgrading. When it is determined that at least one failure has occurred in the network device in step 211, the network device is restarted based on the old version of the software in step 212. Then, the operation returns to step 201. In contrast, when it is determined that no failures have occurred in the network device in step 211, a check is made as to whether the software upgrade is completed in step 213.

When it is determined that the software upgrade is not completed in step 213, steps 210 and 211 are repeated. In contrast, when it is determined that the software upgrade is completed in step 213, the network device is restarted based on the new version of the software in step 214. Then, the operation returns to step 201.

According to the present invention, while software is being upgraded to a new version through a network, if a sudden failure such as a hang-up or latch-up of a network device or a failure in power occurs before the software upgrade is completed normally, the network device is restarted based on an old version of the software so that the network device can attempt the upgrade of the software again without being serviced or without using an external memory device.